Use of Time-Stack Processing in Vehicle Tracking from Roadside 3D LiDAR

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Bachelor of Science with Honors Research Distinction in The Ohio State University

By

Yuyi Chang

Undergraduate Program in Electrical and Computer Engineering

The Ohio State University

2020

Thesis Committee:

Benjamin Coifman, Advisor

Keith Redmill

© Copyright by

Yuyi Chang

2020

Abstract

Light Detection and Ranging (LiDAR) is a perception technology that is widely used in Intelligent Transportation System (ITS) applications. This thesis presents a time-stack based method for tracking vehicles through 3D LiDAR point clouds recorded from a LiDAR scanner installed next to a roadway. The time-stack methodology is adapted from video image processing techniques that sample a 1D or 2D region at fixed time intervals to compile a 2D or 3D "time-stack". The time-stack is a high dimension representation of spatiotemporal data with changing features. For a 1D region, the time-stack is essentially a time-space diagram showing how the visual features evolved through the region over time and space. Various techniques can then be used to find and trace the features in the time-stack in a model-free tracking methodology. In this thesis the time-stack is modified from the conventional video-based format to instead present the evolution of vehicle locations as they travel through the LiDAR field of view.

The scope of this work spans from data preparation to post-processing. The data preparation ultimately identified confounding factors in the data collection, and developed techniques to address these problems. The research developed techniques for ground identification, road boundary identification, vehicle clustering, and vehicle tracking. For a given data scan, the static background was identified and discarded, any remaining data points within the road boundaries were clustered to form vehicles.

A linear road model was introduced as a referencing system for measuring distance traveled along the roads for any vehicle. The processing techniques developed from this research are anticipated to contribute to the development of traffic surveillance and offer valuable insights into driving behaviors.

Dedicated to my grandpa Fukuan,

who sparked my passion in science.

Acknowledgments

First of all, I would like extend my sincere gratitude to my advisor Professor Benjamin Coifman for supervising this thesis. This has been a privilege to work with him, and it is his patient guidance and support to make this thesis possible. I would like to acknowledge Professor Keith Redmill for joining my thesis committee, as well his help during my engineering service activity.

I wish to extend my gratitude to my parents for their support throughout my undergraduate study. It is definitely not an easy task to provide me the opportunity to study aboard. There is nothing I will ever do that can even come close to their love and sacrifice they have made.

I would like to thank Lizhe Li (who soon will become Dr. Li) for his friendship and help. It is his video data reduction tool that inspires me to create a counterpart for LiDAR data, and I am glad I can contribute to his data reduction in my early stage of research. I also thank Zheng Wang for helping me manually processing some of the data. I would like to thank the Civil Engineering Group in the Newcastle University School of Engineering for collecting and sharing the LiDAR data used in this study.

Finally, I thank all my friends, inside and outside of ECE, for sharing this journey. My undergraduate education would not have been as rewarding as it is without you.

This material is based in part upon work supported in part by the National Science Foundation under Grant No. 1537423. The contents of this report reflect the views of

the authors who are responsible for the facts and the accuracy of the data presented herein. This report does not constitute a standard, specification or regulation.

Vita

2016 ..... High School Affiliated to Beijing Jiaotong University

2018-present ..... Undergraduate Research Assistant, The Ohio State University.

Fields of Study

Major Field: Electrical and Computer Engineering

Table of Contents

List of Tables

Table **Page**

List of Figures

Chapter 1: Introduction

Light Detection and Ranging (LiDAR) is a perception technology that is widely used in Intelligent Transportation System (ITS) applications. Although it is commonly used by autonomous vehicles as a collision avoidance sensor, the infrastructure-based application of the 3D LiDAR has not gained much attention. This thesis seeks to develop vehicle identification and tracking methods using recorded data from a 3D LiDAR scanner installed by a roadway.

This thesis presents a time-stack based method for tracking vehicles through 3D LiDAR point clouds recorded from a LiDAR scanner installed next to a roadway. The time-stack methodology is adapted from video image processing techniques that sample a 1D or 2D region at fixed time intervals to compile a 2D or 3D "time-stack". The time-stack is a high dimension representation of time series data with changing features. It is essentially a time-space diagram showing how the visual features evolved through the region over time and space. Various techniques can then be used to find and trace the features in the time-stack in a model-free tracking methodology. This work adapts the time-stack processing to compile multiple frames of LiDAR returns instead of the traditional visual based features, and then develops a methodology to track individual targets through the LiDAR time-stack.

1.1 Background and Significance

LiDAR is a laser-based device that is commonly used as a perception tool for autonomous vehicles. The device sends out a sequence of laser beams over an array of angles and determines the range to a target return at each angle via time-of-flight. This work uses 3D LiDAR that scans a spherical region, yielding a rich picture of how the region evolves over time. Or specifically in the case of this research, after removing the ground features and restricting the search to the region of the road, the 3D LiDAR data provide a rich picture of how the road vehicles evolve.

Vehicle sensing technology has been around for a long period of time. Conventional technology relies on break-beam sensors and loop detectors which can only provide information on about vehicle passing a point in space. Although the introduction of image processing-based traffic surveillance expanded the traffic data collection to include vehicle trajectories, the positioning accuracy is imprecise. The resolution of this technology is vulnerable under different weather conditions, and relative performance is degraded due to camera skewness. LiDAR-based sensing overcomes many of the limitations of image processing, providing precise ranging information under multiple operating conditions.

The LiDAR data processing methods developed in this thesis are significant because it offers a model-free approach to tracking vehicles through 3D LiDAR point clouds. The current practices of LiDAR based vehicle tracking typically rely on using a model to first identify, then segment and then track vehicles. Whereas the time-stack methodology developed here effectively clusters the vehicles in the first step and then these tracks need to be segmented and processed to track the vehicles. This

approach is significant because it could greatly reduce the cost and complexity of LiDAR based vehicle tracking under certain conditions.

The processing methods developed in this thesis could contribute to both theory and practice. First, many ITS applications rely on perception and sensing technologies in order to provide good quality of services. The roadside surveillance helps enhancing situational awareness in the complex urban roadway environment where it is difficult for a single onboard sensor to completely sense the entire surroundings, especially for autonomous driving scenario. The vehicle detection technology could potentially help enable next generation of traffic intersection control, like Adaptive Traffic Signal Control (ATSC) technology. Furthermore, the LiDAR-based vehicle tracking provides high quality of vehicle maneuver behaviors. Traffic flow theorists could benefit from those data to achieve a better understanding in drivers' behaviors.

1.2 Literature Review

Processing 3D LiDAR data consists of multiple steps including background segmenting, target clustering and vehicle tracking. In this work the background segmentation filters remove LiDAR returns from off-road and stationary objects. A statistical-based 3D background filtering algorithm based on common return points was developed by [1] using common return points in each 3D regions. [2] developed a ground identification algorithm using least square fitting and Markov Random Field (MRF). The road model is constructed from identified ground returns, providing measuring reference for a on-road vehicle. [3] proposed road shape estimation methods using returns from multiple LiDAR units.

Vehicle clusters are identified from the points within the region-of-interest. Density-Based Spatial Clustering of Applications with Noise (DBCAN) is the most popular method among clustering algorithms developed in [4], and its distance-based [5] deviation is used in LiDAR scenario where dynamic clustering model parameters were introduced. K-mean is another clustering algorithm often used for vehicle clustering in 3D LiDAR data. Machine learning is an emerging technique that used in background filtering, including Backpropagate Artificial Neural Network (BP-ANN) [5] and deep Convolutional Neural Network (CNN) [6].

Most LiDAR based vehicle tracking algorithms are model-based because the density of returns from a target can rapidly change by several orders of magnitude as the target changes position relative to the LiDAR sensor. [7] developed on-board vehicle detection and tracking using 2D LiDAR and video camera of a probe vehicle. Cluster modelling helps extracting trackable features from a given point cloud. [8] developed a surface modeling method, and [5] introduced a cluster referencing technique using an eigen-point. As different parts of a vehicle may be seen at different location and time, vehicle feature identification and registration is necessary to understand the formation of vehicle point cloud. [9] developed a method to construct vehicle models using local features. Vehicle tracking relates vehicle clusters in time, producing time series position, velocity and acceleration information. [5, 10, 11, 12] proposed tracking techniques via methods including nearest neighbor, Kalman filter, and Multiple Hypothesis Tracking (MTH).

There are a few papers on developing model-free method for tracking objects. A 2D LiDAR model-free implementation is developed by via on-board sensing in [13], and 3D LiDAR based method is introduced in [14] to capture long range targets

via gird-based identification. The similar methodologies are also developed in video-based traffic monitoring. The model-free video processing techniques were developed to track vehicles [15, 16, 17] and pedestrians [18] using CCTV cameras.

1.3 Overview of Thesis

The remainder of this thesis is laid out as follows. Chapter 2 presents the methods needed for vehicle identification, including data collection and pre-processing methods, and the formation of time-stack that is used throughout the thesis. This chapter presents background segmentation to simplify tracking and eliminate off-road targets, and it only retains LiDAR returns that are actually on the road to obtain high quality vehicle clusters with minimal confounding non-vehicle returns. Chapter 3 then develops a model-free approach to tracking vehicles through the time-stack. This chapter also develops a simple linear referencing system to project from the 2D coordinate system in space to a 1D coordinate system along the road. The thesis closes with Chapter 4, that summarizes the work and discusses the conclusions.

Chapter 2: Data Acquisition and Pre-processing for Vehicle Identification

This chapter presents the methods needed for vehicle identification, including data collection and pre-processing, the formation of a time-stack, and background segmentation techniques that only retain the on-road LiDAR returns.

2.1 Data Collection and Representation

This thesis studies a 100 second long 3D LiDAR point cloud file collected on an urban arterial in Newcastle Upon Tyne, United Kingdom, as shown in Figure 2.1. The 3D LiDAR was mounted on a tripod set up next to the road, at a height of approximately 1 m and oriented with a slight tilt to the ground plane. This data set includes a bus pulling up to a stop and then proceeding, as well as 15 cars, and several pedestrians crossing the road.

A VLP-16 3D LiDAR module, manufactured by Velodyne as shown in Figure 2.2, was used for data collection. The LiDAR unit scans a 360° horizontal Field of View (FOV) with a 30° vertical FOV and maximum detection range of 100m [19]. The sampling rate was set to 10Hz, so the entire data file consists of 1000 frames. As indicated by its name, the sensor emits sixteen laser beams at each azimuth, providing vertical scan angles between -15° to 15°.

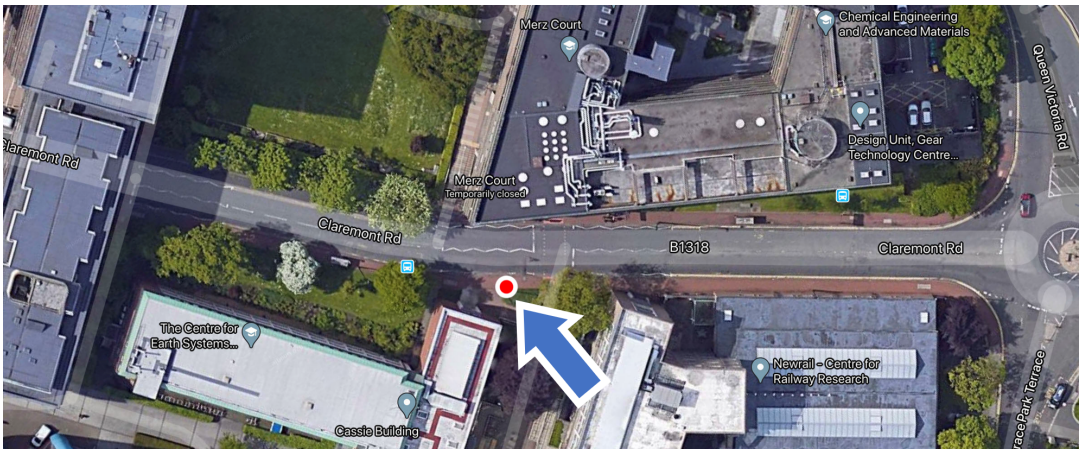

Figure 2.1: Satellite image of Claremont Rd. where LiDAR data was collected, capturing 16 vehicles and several pedestrians. Collection location is marked by a red dot. Image from https://www.google.com/maps

Figure 2.2: A Velodyne VLP16 3D LiDAR unit. Image from https://velodynelidar.com

The collected data, stored in .pcap format, was converted into .mat file format to facilitate loading the data into MATLAB for analysis. Appendix A provides more details on the .pcap package structure of the data file. The LiDAR data recorded in a spherical coordinate system, which is converted a Cartesian coordinate, using the techniques from [19] as shown in Figure 2.3.

The MATLAB Computer Vision Toolbox was used to perform data format converting. The data package was loaded using utilities provided in Lidar and Point

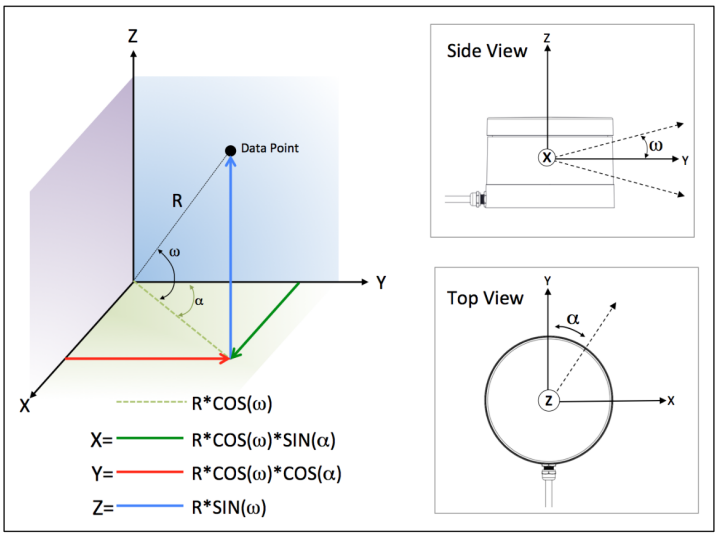

Figure 2.3: Converting spherical coordinate to Cartesian coordinate [19]

Cloud Processing package, where the data is managed by a data loader. To obtain LiDAR scan at a given frame ID, the data loader returns a *pointCloud* class object whose location data is coded in a 3D sparse matrix of size $m \times 16 \times 3$. The first dimension m represents the number of azimuth scans, which varies from frame to frame. The second dimension matches the sixteen vertical scan angles (elevations) the sensor unit has. The third dimension maps to x, y and z coordinate layers in Cartesian coordinate. Any scan angles with no returns are filled with Not-a-Number (NaN).

A MATLAB-based graphic user interface application that we call LidarVisualizer was de- signed to facilitate visualization and data reduction from the raw LiDAR data. The application provides functionality that include principle views display (front, side, right and isometric), frame aggregation/comparison, manual segmentation and clustering. Details of this application can be found in Appendix B.

2.2 Data Pre-processing

2.2.1 Frame Correction and Recovery

The original data contained two confounding factors that negatively impact the accuracy of the processing results. The first was a tilted angle because the LiDAR sensor was not parallel to the ground plane, as shown in Figure 2.4.a. A correction is necessary to facilitate the ground identification and removal algorithms, which assume that the ground is flat (i.e. parallel to xy plane). We corrected the angle by applying a pure-rotation affine transformation where the rotation angles were found manually.

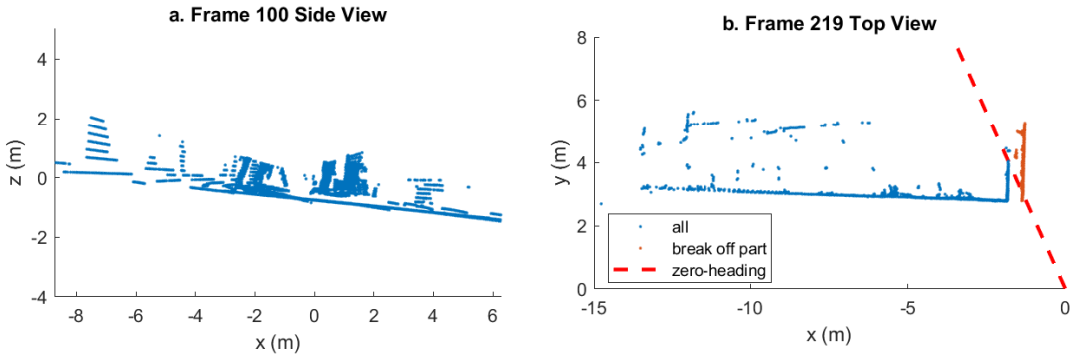

Figure 2.4: Tilted scan angle (left) and incorrect zero-heading(right)

The second confounding factor addressed in the pre-processing stage is the fact that the zero-azimuth was pointed to the region of interest, but the sensor advances the frame at zero-azimuth, which lead to temporal inconsistencies in the scans. The LiDAR sensor used in this study reports data continuously in 360° rotation, and a single scan is reported with the azimuth successively ranging from 0° to 359.99°. The original images showed cut-off effect for target returns span across zero-heading

location where the points on either side of the 0° angle were sampled a full sampling period apart, as shown in Figure 2.4.b. The 0° angle is shown for reference with a solid line radiating from the LiDAR sensor. Notice the vertical red line to the right of the 0° angle should continue the vertical blue line to the left of the 0° angle. In the original data set all returns to the to the right of the 0° angle were collected one sample period before the data immediately to the left of the 0° angle. To address this problem we resampled each frame by adding one time step to all returns to the right of the 0° boundary, effectively resetting the zero-heading end of cycle time jump to an azimuth outside of our region-of-interest.

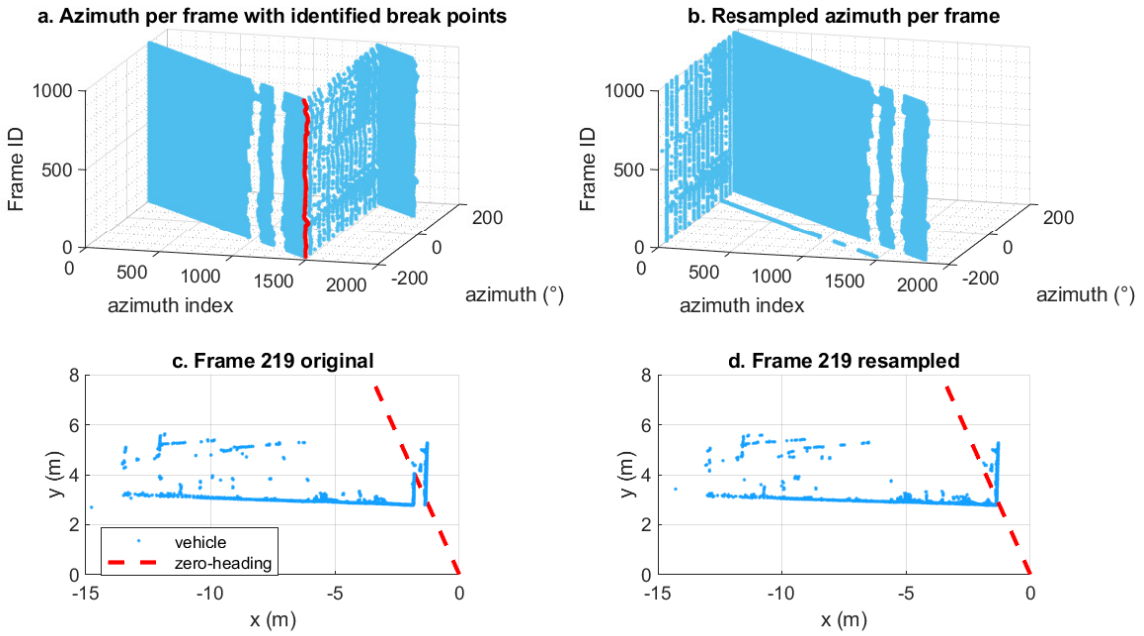

Figure 2.5: Azimuth for all frames with identified break points in red

We resampled the LiDAR scan to remove target break-off as seen in Figure 2.4.b with zero-heading direction shown in dash line. For each frame, azimuth is indexed

with sample time, in which the sample return time stamp increases linearly with azimuth index. Hence, a jump in azimuth as seen in Figure 2.5 at azimuth index of 1500 suggests the continuity of azimuth is interrupted. As the LiDAR samples 360°, sample azimuths are both 30° on each end but separated about 2000 azimuth index about a full sample second. Therefore, the 0.1 sec sampling time difference, combined with vehicle motion, results breaks in vehicle back scan in Figure 2.4.b. Using the azimuth jump as the new start point of each sampling window, the new LiDAR scans were reconstructed.

Properly setting up the LiDAR sensor facilitates pre-processing efficiency. If a tilted angle is desired, it is possible to record angle of installation and rotate the sampled frames to a position parallel to true ground plane when analysis the data, for which ground identification algorithm like [20] requires horizontal mounting angle. The cut-off effect can be avoided by pointing the zero-heading line of the 3D LiDAR to an off-road region where no target-of-interest present. Yet, this may not be possible if the scanner is used surveillance the entire 360° horizontal FOV, like installing the LiDAR at the center of an intersection.

2.2.2 Time-Stack Construction

A time-stack is a collection of spatial information over a time interval. In this case each layer of the stack corresponds to the XY coordinates of all of the on-road returns, and one layer is recorded for each frame of the LiDAR point cloud. The selected features were recursively placed into the time-stack, forming a high dimension sparse matrix. Any sampled angles in a given frame with no "on-road" returns were represented as Not-a-Number (NaN) in the time stack. The entire duration of the

LiDAR data set was used to construct the time-stack for two lanes at the study location.

The time-stack as a powerful tool in preprocessing and identifying vehicles in the LiDAR point cloud. Discretization is necessary for data spans in a wide range of uncertainty. Here, sparsity of data determines effectiveness of a time-stack as there are few samples in each category to make decision based on group statistic. The reported azimuth angle values were on the order of 0.01°, representing a nearly continuous variable over the range from 0 to 360° range. To achieve meaningful results it is necessary to bin the azimuth angles to construct the time-stack. In this thesis, we employed a static time-stack with azimuth angles rounded to the nearest 0.5°

2.3 Vehicle Identification

This section presents the methodology used to extract the vehicles from the point cloud. Decisions for processing and analysis were made based on characteristics and statistics of the point cloud. The work sought a balance between manual and automatic processing where manual processing is an time-efficient approach in some situations as we only focus on a single dataset in this thesis.

2.3.1 Background Filtering and Ground Segmentation

Background filtering via road boundaries

A single frame contains thousands of data points, but only a few hundred of them represent targets of interests to us. To vehicles in a given frame, this work uses background filtering and ground segmentation to remove the non-vehicle returns. In the first step, given a LiDAR data sequence where the sensor is fixed, we manually defined the road boundaries by inspecting the point cloud, in particular plotting all

of the frames overlapping in a single image. After defining the road boundaries the sample returns located inside of the defined road boundary were kept and we call this the set of the "on-road" samples, and those outside of the road boundary were excluded from further processing. Individual lane boundaries were defined within the road boundary. The current dataset contains two opposing lanes where we refer the near lane and far lane based on their relationship to the LiDAR sensor.

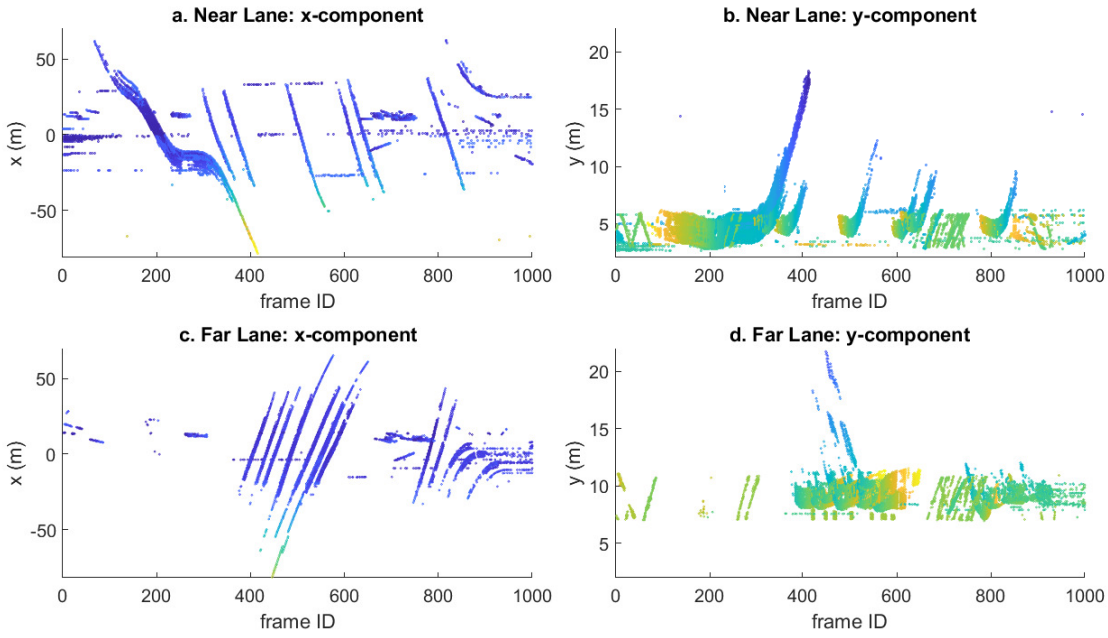

Figure 2.6: Tima-stack containing vehicle, pedestrians and noises

Figure 2.6 shows the time-stack for near and far lanes, showing x and y components separately. The x components are shaded using the magnitude of y components, and vice versa for the shading of the y components. In Figure 2.6.a, the wide track moving from $x = 50$ to $x = -50$ in frames 50 to 400 is a bus that pulls to a stop, and then six subsequent vehicle tracks sweep through the same x range. The span of x and y

are determined by road geometry. In this dataset, distance travel along x is larger than along y as the road is generally extending in x direction. Yet, we see short cyan lines in part d of the figure, which corresponds to pedestrians crossing. As most of pedestrians move perpendicular to the road, we are able to see a short x direction span in part a and c of the figure at around $x = 20m$. Although the automatic processing removed most of ground returns, there are some background clutter left as seen as horizontal lines exist throughout the frame duration.

Ground segmentation

At this point the on-road samples include returns from both moving and fixed objects. The moving objects include vehicles and pedestrians, while nonmoving objects include the road pavement and trees overhanging the roadway. We developed the *angle-of-no-return* algorithm for ground removal. The algorithm input requires a time-stack consisting range information indexed with azimuth θ, elevation ϕ and frame ID i. The scan angle (azimuth and elevation) with no returns are indicated by NaN. For each scan angle pair, the algorithm determines ground (fixed) and non-ground (moving) points within a time-stack based on percentage of Non-NaN returns and most commonly seen range. The percentage tolerance for valid return is a variable of choice. The span of frame IDs, i, is set be long enough to avoid the situation where a stopped vehicle is might incorrectly be classified as ground returns.

Algorithm 1: Angle-of-No-Return Algorithm

Input: Range time-stack $R[\theta, \phi, i] = \{R_1, R_2, \ldots, R_n\}$
Output: Logical array $isGround$
Initialize $isGround = false(size(R))$
Initialize percentage error tolerance ϵ
for $\theta \leftarrow 1$ ***to*** end **do**
  for $\phi \leftarrow 1$ ***to*** end **do**
    Compute percentage of Non-NaN return, η
    $\beta \leftarrow mode(R[\theta, \phi, :])$
    Test z-threshold
    if $1 - \eta < \epsilon$ **then**
      $isGround[\theta, \phi, argwhere(R[i]\ deviates\ from\ \beta)] = False$
  end
end
return $isGround$

Non-moving objects were removed in individual frames, and a time-stack was formed based on the remaining moving objects. The result time-stack contained vehicles and pedestrians that were within the region-of-interest.

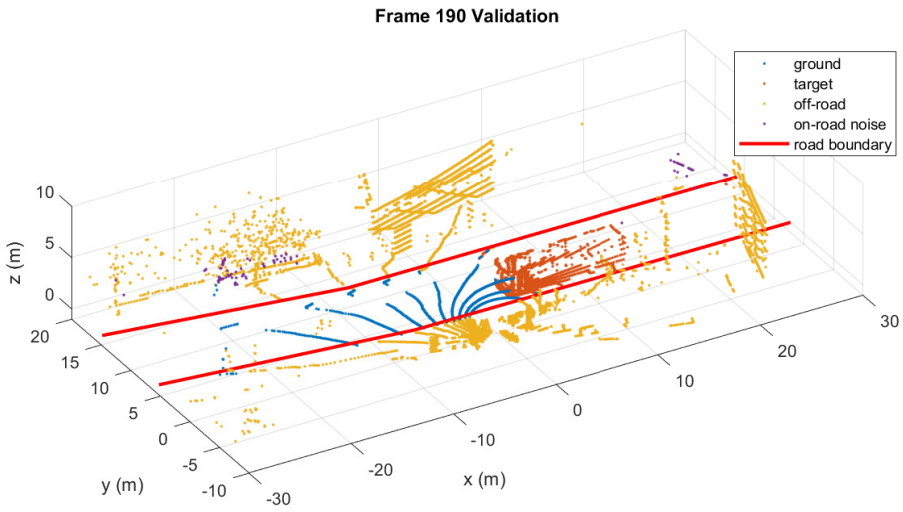

Figure 2.7: Single frame validation after background filtering and ground segmentation

Figure 2.7 shows a single frame from the validation dataset where ground, off-road, target and noise returns are marked in different colors. The red lines denote the manually extracted road boundaries. By excluding the off-road returns, shown in orange, we significantly reduce the volume of data as the majority of returns are from off-the-road, e.g. nearby buildings. For the on-road region, the ground plane, identified by *Angle-of-No-Return* Algorithm, is marked in blue. The remaining parts, marked in red, contain target vehicles, which is our objective. A few non-vehicle returns are remain in the time-stack (e.g. the purple clutter at $x = -20m$ and $y = 16m$ are returns from trees). So all returns from this region were manually suppressed since the trees were high enough off the road that they could not have come from a vehicle. We performed manual single frame validation to confirm the segmentation results.

2.3.2 Channel of Detection

We performed vehicle clustering based on the time-stack consisting of the on-road returns. Based on the sensing environment, we used the following criteria.

1. Since the only points of entry and exit are the extremes of the observed roadway, based on conservation, a vehicle exists throughout the range of detection as long as it is seen once within that range;

2. A vehicle satisfies the previous assumption strictly within the given lane's time-stack since there are no Lane Change Maneuvers (LCM) in the dataset;

3. A path of a given vehicle can be approximated by a linear model.

The first two criteria aim to solve the problem of occlusion. For example, a near lane vehicle blocks the FOV of our LiDAR sensor so that the far lane vehicle is not visible for a period of time. The first criterion is based on the principle of conservation where a vehicle instance cannot disappear except at the exit of the detection zone. The second criterion incorporates First-In-First-Out (FIFO) principle where vehicle queueing does not disrupt the sequence of the detected vehicles. The third criterion, aims to address the occlusion situation where we estimate a vehicle's future path for a short period by extending its known speed and position linearly forward in time.

2.3.3 Vehicle Clustering

The automatic vehicle clustering was designed to identify individual vehicle instances presented in the time-stack for a given lane. The clustering decision was made based on ranges of x, y, and frame ID a vehicle occupies within a given time-stack. The time-stack, consisting of coordinate information with frame ID, is discretized into individual coordinates and frame ID bins. Each bin stored a binary indicator as to whether or not it was occupied in the given frame. A 2x2 logical matrix of all-true was applied to the previous binary image via 2D convolution in order to remove noise and expand the existing vehicle trajectory regions. This convolution kernel and the 2D convolution achieve both dilation and erosion, which are commonly used in image processing for noise reduction. The resulting contours determine the region-of-interests of individual vehicles in time and space. This process was repeated for both the near and far lanes.

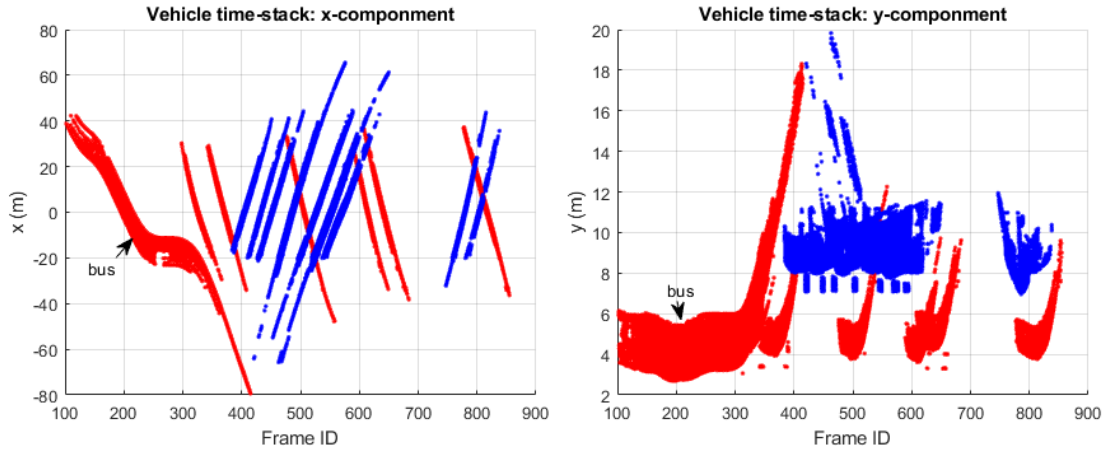

Figure 2.8: Vehicle time-stack of near lane (red) and far lane (blue)

Vehicles in the Time-Stack

Figure 2.8 shows the time-stack consisting of only the vehicle returns for the near and far lanes. The x and y components of the vehicle time-stacks are shown separated to facilitate visualizations. Besides the range of travel in each direction, occlusions are visible where the near lane vehicle passes "in front" of a far lane vehicle. On the left plot, occlusions are evident at frame 480, 620 and 800 where the far lane vehicle trajectories (in blue) briefly disappear in the FOV and reappear after a few frames. It is clear that a vehicle cannot disappear or show up within a FIFO road as mentioned in section 2.1.4. Two disjoint vehicle clusters can be related by considering time-dependency of vehicle motion. A single frame validation at frame 508 is shown in Figure 2.9 where multiple vehicles are correctly clustered. Vehicles A - C are in the far lane, and vehicle D is in the near lane. The blue dash lines on the two lower subplots show the instant in the time-stack corresponding to the frame at the top of the figure. Although vehicle A, B and D are all cars as inferred from the span on

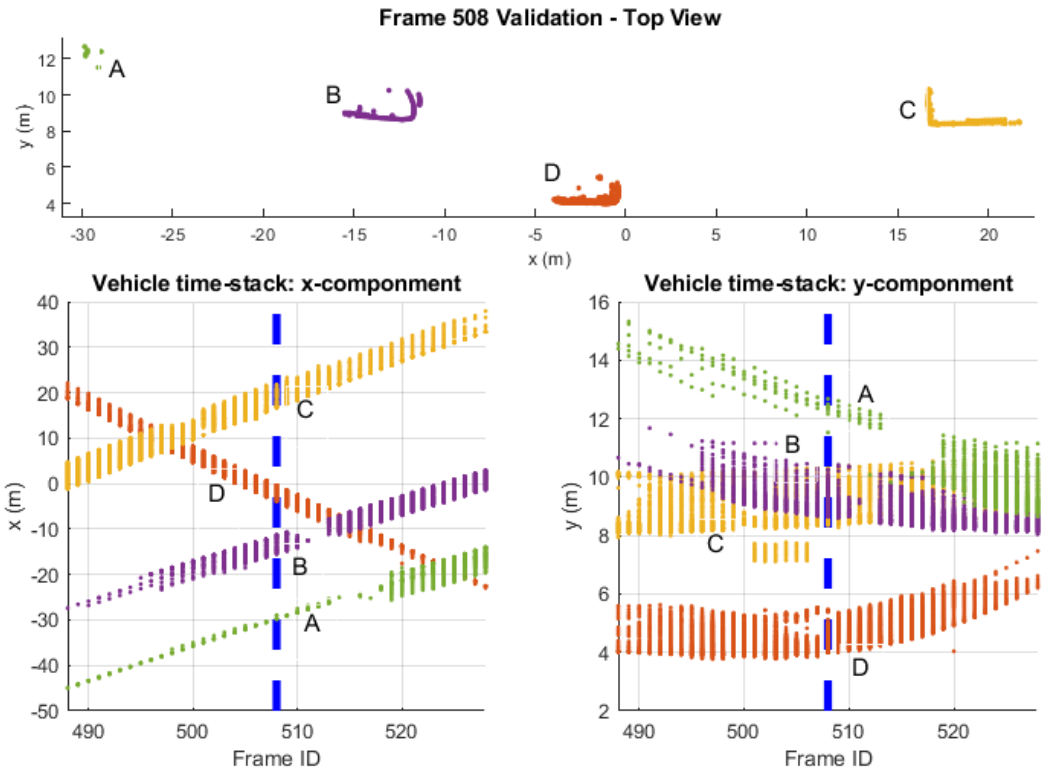

Figure 2.9: Clustering validation frame 508. Vehicles A-C are far lane vehicles, and vehicle D is from near lane. The blue dash line denotes frame where the top view is obtained

the time-stack, vehicle A has significantly fewer returns compared to vehicle B and D, which are close to the LiDAR sensor.

Figure 2.10 shows all of the vehicle tracks from the near lane in the XY plane, where each vehicle is marked in a unique color. Since traffic in the UK drives on the left-hand side of the road, vehicles in the near lane are travelling toward the negative x direction. One track, shown in blue, pulls to the left of the rest of the vehicles in Figure 2.10. This track is from a bus that pulled over for a bus stop downstream of the LiDAR sensor (located at 0, 0), joining back to main lane at $x = -35m$. The

stopped bus is also evident in Figure 2.8 with a constant x and y coordinates between frames 260 and 300.

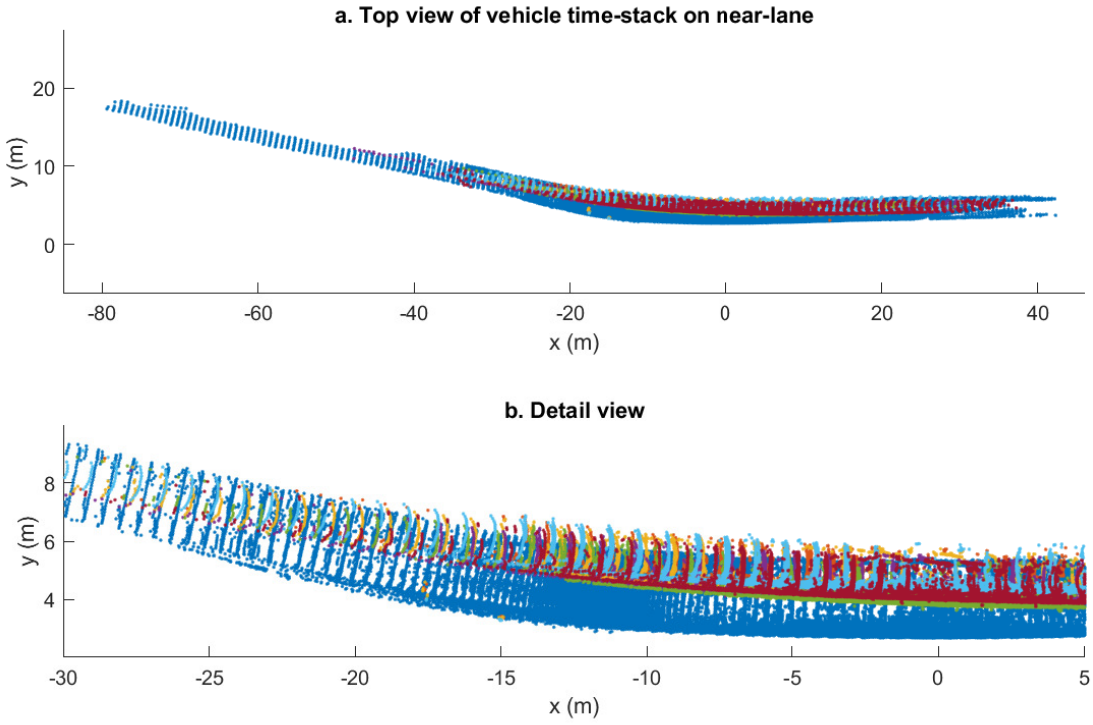

Figure 2.10: Top view of the near lane vehicle time-stack

Chapter 3: Model-free Vehicle Tracking using Time-Stack

This chapter presents the model-free vehicle tracking using the time-stack as developed in this research.

3.1 Road Referencing Model

The first step in the vehicle tracking is the construction of linear road model to quantify the longitudinal distance along the road.

3.1.1 Distance Measurement along Road Coordinate System

A road coordinate system provides referencing framework for vehicles driving in the same lane where the spatial occupancy can better be represented on a single referencing axis, and the model provides projection reference to measure vehicle travel distance along the road. The representation enables distance measurement along the road where a higher dimension is projected down to a few degrees-of-freedom. A vehicle travelling on a real world road has six degrees-of-freedom: x, y, z, roll, pitch and yaw. We are interested in projecting this high dimension representation to a 1D road coordinate system. We defined a s-coordinate notation where the distance along the road centerline, denoted as s, is a function of real world coordinate of road.

We inferred road curvature, θ, from the 2D geometry of the road. In this way, we constructed a macroscopic vehicle spatial representation as per Equation 3.1.

$$Macro(t) = [s(t),\ \theta(t)]^T. \tag{3.1}$$

From [21], the microscopic relationship between a vehicle's center and the road centerline is described as a shift in lateral position and heading, denoted as Δl and $\Delta\theta$. Since it is conventional practice to treat the road as a flat surface, we dropped the z-component by only considering vehicle position in x, y and heading, which is the microscopic vehicle spatial representation given by Equation 3.2.

$$Micro(t) = [Macro(t),\ \Delta l(t),\ \Delta\theta(t)]^T. \tag{3.2}$$

3.1.2 Linear Road Model

We propose a linear road model for measuring distance along a given road. The method breaks a curvy road into multiple linear segments where vehicle returns are projected into different corresponding regions. The order of the model is scalable to achieve better approximation. The model is designed to use either GIS data or manually drawn road data. The more accurate the road data are, the better the approximation from the road model.

Linearization of actual road

To start our derivation, we first consider an actual road that exits in real life. We generally assume the road is flat, for which a curvy region with high elevation changes is not ideal for installing LiDAR sensor. We take the centerline of the road, which can

be defined as a curve γ. We parametrize the curve such that $\gamma(s) = (x(s), y(s))$ where the z component of the curve is dropped, and x and y are differentiable functions.

We approximate this curve via First Order Taylor Expansion at real number a where

$$\gamma(s) = \gamma(a) + \frac{\gamma'(a)}{1!}(s - a) \tag{3.3}$$

Hence, we can pick a start point $P_0 \in \gamma(s)$ and approximate another point P_1 on the curve as $P_1 = P_0 + \gamma'(P_0) \cdot (P_1 - P_0)$. Suppose the line segment between points P_0 and P_1 is a sufficiently good representation of actual road curvature $\gamma(s)$, we call two points as *nodes* of the road. Any point $p \in P_0P_1$ can be obtained via linear interpolation of line segment P_0P_1.

Now, we can draw a sequence of nodes from $\gamma(s)$ as $P = \{P_0, P_1, \ldots, P_n\}$, and we call the line segments formed by this sequence of nodes as linear representation of road, or *linear road model*.

Road node, normal, direction and distance

A skeleton of road is represented by directional nodes $\{P_0, P_1, \ldots, P_n\}$ as discussed in previous part. Now, for each linear road segment formed by $\boldsymbol{r_k} = P_kP_{k+1}$ where $k \in [0, n-1]$, we define unit normal vector $\boldsymbol{n}$ and unit tangent vector $\boldsymbol{m}$ of the road segment as

$$\boldsymbol{m} = \frac{\boldsymbol{r_k}}{\|\boldsymbol{r_k}\|},\ \boldsymbol{m} \cdot \boldsymbol{n} = 0. \tag{3.4}$$

For a point $\boldsymbol{p}$ on the road segment P_kP_{k+1}, we can represent the point as a linear combination

$$\boldsymbol{p} = \boldsymbol{P_k} + s_k \cdot \boldsymbol{m} \tag{3.5}$$

where s_k represents distance along the segment. Hence, for any $k \neq 0$, the total distance along the road from initial node P_0 is addition of segment norms.

$$s = \sum_{k=0}^{k-1} \|\boldsymbol{r_k}\| + s_k. \tag{3.6}$$

3.1.3 Road Model Construction

Figure 3.1 shows road model established from near lane vehicle time-stack in Figure 2.10 and the aerial photo in Figure 2.1.

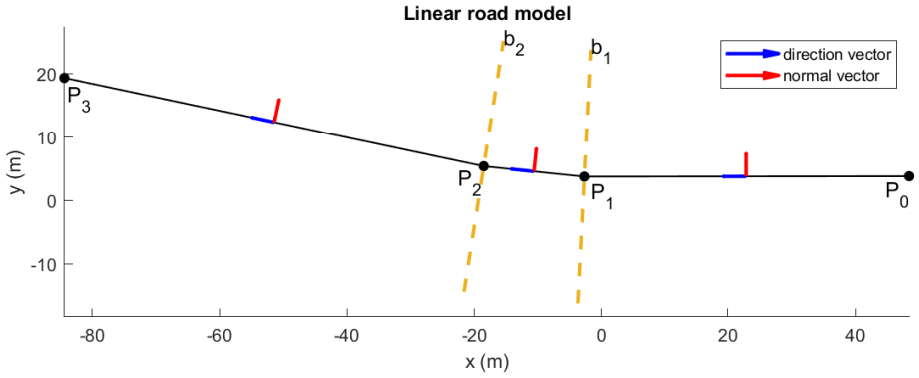

Figure 3.1: Linearized road model used in this thesis

The model consisted of four nodes, forming a directional path from P_0 to P_3. The result three sectors are separated by two bisector vectors b_1 and b_2, and road tangent and normal vectors are shown in blue and red, respectively. The road segment P_1P_2 is to approximate road curvature as the actual road slowly turns upward in the y direction. Although the model was derived from near lane vehicle aggregated frames, this referencing system is used for both near and far lane vehicles.

3.2 Vehicle Position Tracking

The time-stack of each lane containing all vehicle point clouds were projected to the road model. The projection maps the x and y coordinates of the returns to a distance along the road where a vehicle position along the road can be represented by a range of s.

3.2.1 Bisector and Sector attribution

We use bisector vector to represent boundary of two adjacent road segments. A bisector vector $\boldsymbol{b_k}$ for road segment $\boldsymbol{r_k}$ for $k > 0$ is defined as

$$\boldsymbol{b_k} = \frac{\boldsymbol{r_k} - \boldsymbol{r_{k-1}}}{\|\boldsymbol{r_k} - \boldsymbol{r_{k-1}}\|}. \tag{3.7}$$

Bisector vectors define point-segments mapping relationship where any point not on road segments are projected to corresponding road segment if the point is within the sector of the segment. To determine the point attribution, we design a search algorithm for given points.

Algorithm 2: Nearest Sector Search

Input: Sample returns $\boldsymbol{p_1}, \boldsymbol{p_2}, \ldots, \boldsymbol{p_m}$, Road nodes $\boldsymbol{P_1}, \boldsymbol{P_2}, \ldots, \boldsymbol{P_n}$, road bisector vector $\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_n}$
Output: Sector index vector $\boldsymbol{sectID}$
Initialize $\boldsymbol{sectID} = false(m, 1)$
Initialize $\boldsymbol{isSet} = false(m, 1)$
for $i \leftarrow 1$ ***to*** n **do**
  $\boldsymbol{\theta} \leftarrow$angle$(p - \boldsymbol{P_i})$
  Sector angle range $angRng \leftarrow$ angle$(\boldsymbol{b_i}, \boldsymbol{P_i})$
  for $j \leftarrow 1$ ***to*** m **do**
    if $\theta[j] \in angRng$ & $!isSet[j]$ **then**
      $sectID[j] \leftarrow i$
      $isSet[j] \leftarrow True$
  end
end
return $isGround$

3.2.2 Vehicle Point Cloud Projection

The LiDAR returns are orthogonally projected onto the corresponding road segment. Given a sample point $\boldsymbol{p_j}$ locates in sector k, the projection is given by

$$s_k = (\boldsymbol{p_j} - \boldsymbol{P_k}) \cdot \boldsymbol{m_k}. \tag{3.8}$$

The total distance along the road can be calculated using Equation 3.6. We can offset the total distance measurement so that the zero distance is at the closest point on road from the LiDAR position (the origin).

Figure 3.2 shows vehicle time-space diagram for both lanes as a result of an orthogonal projection from the point cloud to the road model. Relative distances of road nodes are marked on the plot, and the total distances are offset to have $s = 0$ at the point closest to the LiDAR scanner. Through projection, the x and y locations are combined into a single dimension s showing distance of travel along the road. Similar to Figure 2.6, occlusions in the far are evident in this figure.

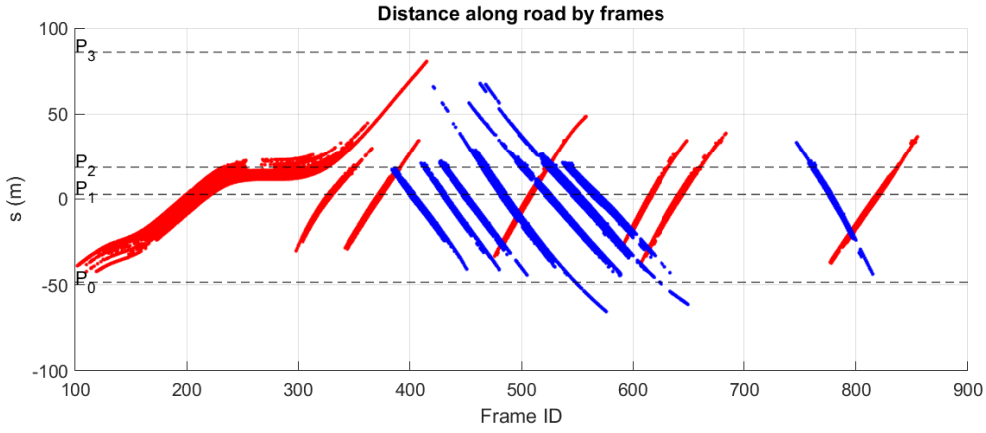

Figure 3.2: Vehicle projection results for near lane (red) and far lane (blue)

Figure 3.3 presents an occlusion situation where a near lane vehicle blocks a far lane vehicle to be seen from the LiDAR. The top view of vehicle point clouds are shown at the time before, during and after occlusion happens. As two vehicles meet at the same azimuth angles, we only see near lane vehicle, shaded in red, where no target is observed in far lane between the space within pre- and post-occlusions, shaded in blue and yellow. Similarly, no distance along the road measurement is observed for the far lane vehicle during the occlusion.

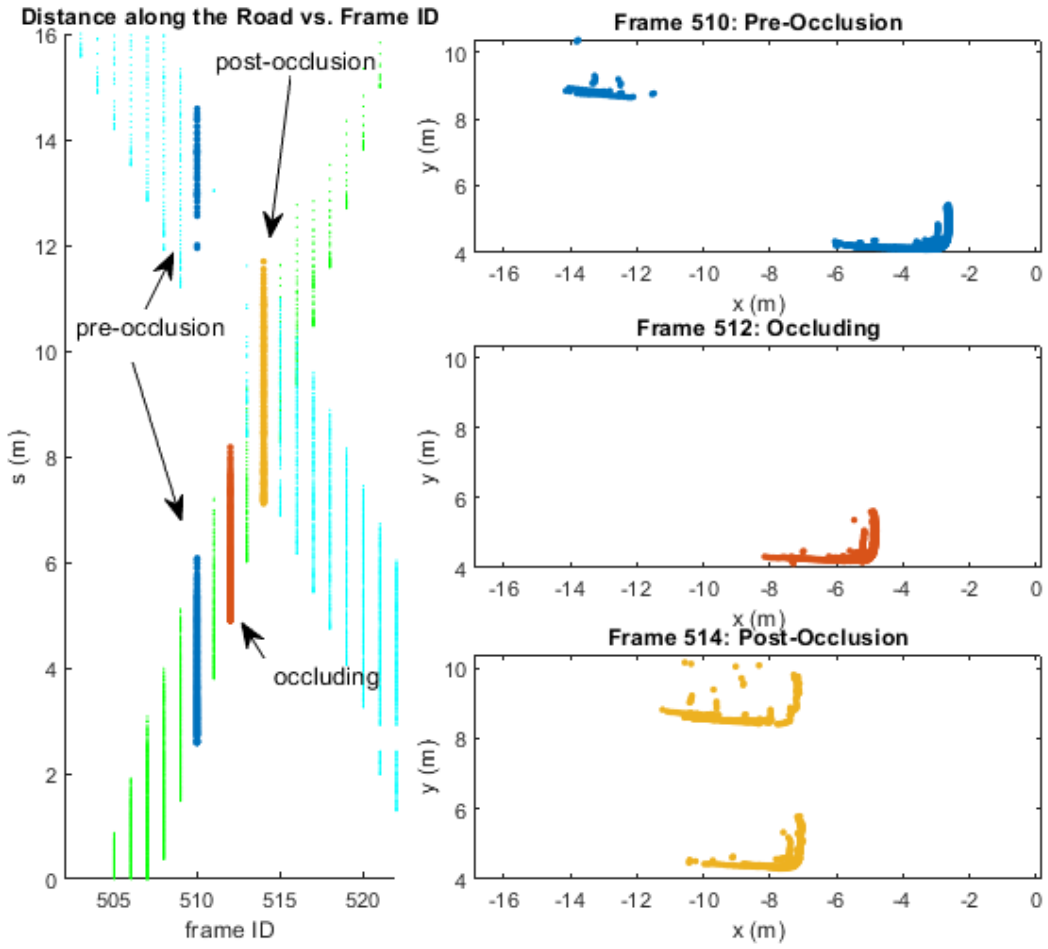

Figure 3.3: An occlusion scenario at frame 518

The maximum detection range varies for different types of vehicles. For buses, a large range of detection between -45 to 80 m is achieved. Yet, the maximum possible detection ranges drops to -40 to 20 m for cars due to the shorter height.

The range of vehicle trajectories reflects the resulting range of detection for the different vehicles. The detection ranges are asymmetric between the downstream and upstream directions for two reason. First, the mounting angle of LiDAR is tilted to the ground, resulting more scan lines striking negative s direction, which contribute to the most of detection range. The range of detection is also affected by the road geometry, which includes both curvature and elevation. As seen from the road model, the road extends mostly in the x direction, which corresponds to the negative s portion. Yet, the road starts gradually turning in the positive y direction in positive the s portion. The impact of road elevation changes is trivial as the road covered in this study is generally flat.

The range of tracking is less than the range of detection due to the fact that the number of return rapidly drops as it travels further from the LiDAR sensor. As can be seen from 3.2, the range of s shrinks as vehicles move away from the sensor.

3.2.3 Vehicle Geometry Understanding

We extracted vehicle geometry information from the vehicle time-stack where the vehicle length and most dominant features were identified. The side of a vehicle seen from the LiDAR is determined by the relative location between the sensor and the target. We picked the most dominant vehicle parts seen in the time-stack to determine the location of a vehicle, as summarized in Table 3.1. The front bumper is visible when the vehicle is moving towards the LiDAR, which maps to the negative

s and positive s portions for near and far lanes vehicles, respectively. Similarly, the rear bumper is seen by the scanner when a near lane vehicle is departing from the scanner, or an approaching far lane vehicle.

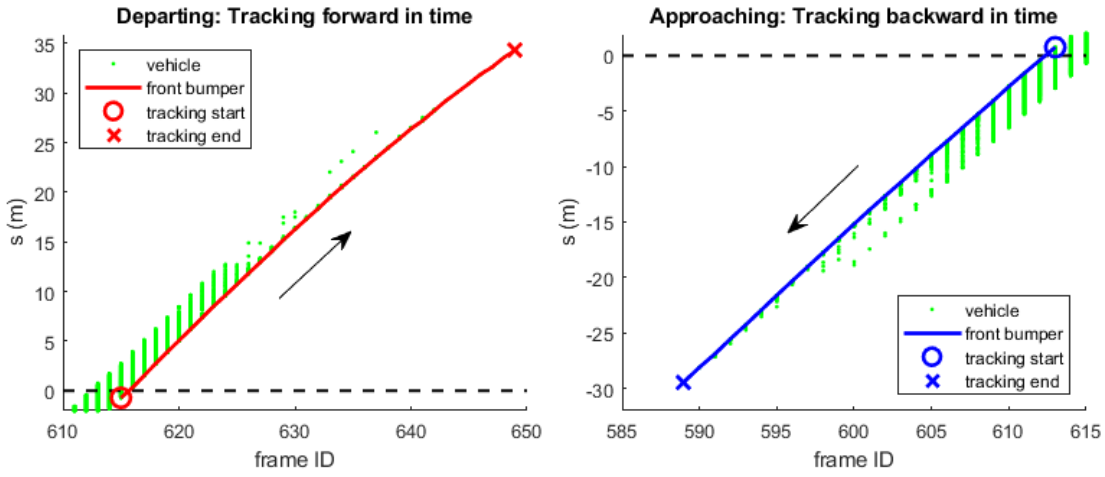

Figure 3.4: Front/Rear bumper tracking at different locations

Figure 3.4 shows tracking a near lane vehicle at different locations where the front and rear bumpers are shaded in red and blue, respectively. We start tracking the rear bumper of a vehicle as it departs from the LiDAR sensor where the full rear surface of a vehicle is within the LiDAR FOV the moment it passes the road origin ($s = 0$), and the tracking direction is shown in an arrow. The starting frame of tracking, indicated by an "o" on the plot, is extended a few meters to create a hand-off buffer zone for tracking. As the vehicle drives away, the total number of returns drops, but of these the percentage of rear surface point returns increases compared to the overall number. We keep tracking the minimum s of the projected vehicle cluster, forward in time, until the vehicle exits from the detection range, marked by a "x" in the

figure. Then, we track the front bumper using the same technique except in tracking is performed reversed in time. Starting at the last moment where the front surface is visible, we track the maximum s of a near lane approaching vehicle backward in time until the vehicle disappears in the range of detection. We repeat this process for far lane vehicles.

Finally, to relate the front and rear bumper tracks form a given vehicle, we developed a map to each end of the s measurements for the vehicle, which are obtained by taking the maximum and minimum of s for the given cluster in each frames. The edges of front and rear bumpers are simultaneously visible as the vehicle passes the LiDAR scanner. The predicted location, determined by s positions at each end, varies based on the scan resolution. The length of each vehicle was determined by the maximum span of s from all frames observed. In this way, we assume all vehicles are travelling precisely along the road direction. Using the criterion provided in Table 3.1, projected locations of front and rear bumpers were determined. These are then used to extend both the front and rear trajectories for the period of time while the vehicle is adjacent to the LiDAR sensor.

Table 3.1: Vehicle dominant features in different relative positions

Relative Location	Dominant Feature	Sign of s	
Approaching	Front bumper	Near lane	-
		Far Lane	+
Departing	Rear bumper	Near lane	-
		Far lane	+

Figure 3.5 and 3.6 show the vehicle front and rear bumpers identified from time-stack vehicle tracks. The front bumpers, shaded in blue, are seen when a vehicle is approaching the LiDAR. When the vehicle departs from the LiDAR, rear bumpers, shaded in red, are visible until a given vehicle exits from detection range. Table 3.2 summarizes the vehicle length estimated from range of s. The order of vehicle ID represents the sequence seen in within the corresponding lane.

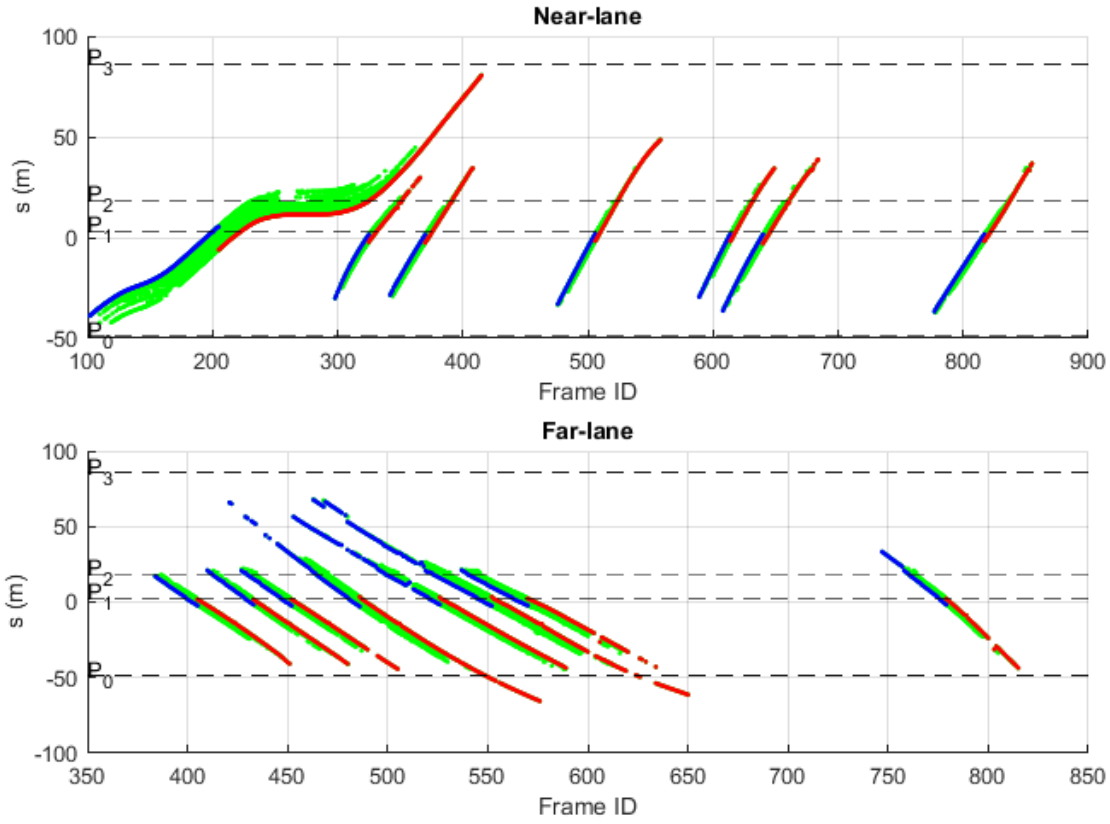

Figure 3.5: Vehicle front (blue) and rear (red) surfaces as seen from vehicle tracks

The difference between vehicle and road orientations could contribute to projection error when measuring the vehicle space occupancy using the s boundary. Since we

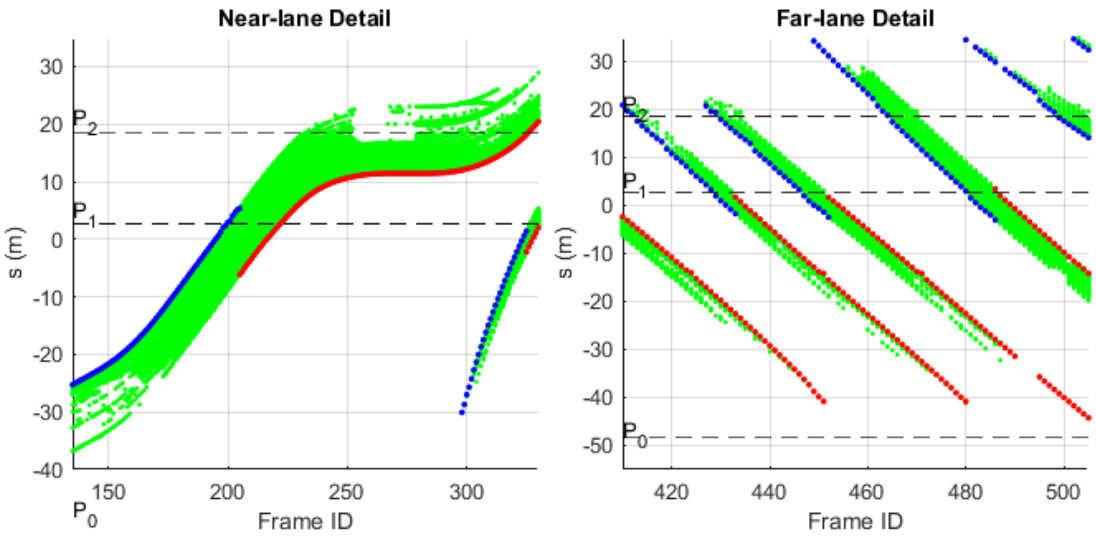

Figure 3.6: Detail view of vehicle front (blue) and rear (red) surfaces

Table 3.2: Estimated vehicle length

Near lane	Vehicle ID	1	2	3	4	5	6	7	
	Length (m)	12.1	3.7	3.9	3.7	3.8	4.6	3.5	
Far lane	Vehicle ID	8	9	10	11	12	13	14	15
	Length (m)	4.8	4.0	5.1	7.0	5.5	7.1	6.0	5.4

use the lateral center bumper positions to describe the vehicle position information along the road, the difference between the actual vehicle heading and the orientation of the road could introduce an additional projection distance for the bumper corners, overshooting the lateral center point of the bumper projections. The vehicle width, W, and heading offset, $\Delta\theta$, can be used to compensate this error where the amount of projection overshoot, Δs, is given by Equation 3.9

$$\Delta s = \frac{W}{2} \sin \Delta\theta \tag{3.9}$$

Chapter 4: Conclusion

This thesis presented a roadside 3D LiDAR based vehicle identification and tracking methodology using time-stack techniques. Chapter 2 presented the collection of the LiDAR data and developed various correction methods to eliminate confounding factors in the data. The chapter also developed a background filtering algorithm to remove off-road and non-moving objects from the data, which enables vehicle identification via automatic clustering. In Chapter 3, we designed a road model based vehicle tracking method where a linear road model was proposed to facilitate understanding of vehicle travel distances along the road. One challenge in this work is the hand-off as a vehicle transitions from approaching the LiDAR sensor to receding from the LiDAR sensor. In between the vehicle is traveling orthogonally to the sensor beams. These changing views require special handling to maintain tracking fidelity. Using the distance measurements obtained from vehicle to road projection, the geometry of a given vehicle was determined. Vehicle geometry was studied from dominant features seen at each location, combing with vehicle length as seen from maximum span of vehicle tracks. The processing techniques developed here are expected to contribute to the development of traffic surveillance and related transportation applications.

There is more research to be done. We hope to automate the process of segmenting vehicles, measure speed and acceleration of all of the vehicles, and many other

logical extensions. To date, this thesis focuses on studying a 1000-frame LiDAR data sequence. Additional data will be examined using these methods. The algorithm for road model optimization will be developed in order to achieve better fidelity when using the time-stack tracking method. We will also consider ways of automatically deducing the lanes of travel from the moving vehicle tracks.

Bibliography

[1] J. Wu, H. Xu, Y. Sun, J. Zheng, and R. Yue, "Automatic background filtering method for roadside lidar data," *Transportation Research Record*, vol. 2672, no. 45, pp. 106–114, 2018. [Online]. Available: https://doi.org/10.1177/0361198118775841

[2] Z. Zhu and J. Liu, "Graph-based ground segmentation of 3d lidar in rough area," in *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, April 2014, pp. 1–5.

[3] K. Peterson, J. Ziglar, and P. E. Rybski, "Fast feature detection and stochastic parameter estimation of road shape using multiple lidar," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 612–619.

[4] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[5] J. Zhao, H. Xu, H. Liu, J. Wu, Y. Zheng, and D. Wu, "Detection and tracking of pedestrians and vehicles using roadside lidar sensors," *Transportation Research Part C: Emerging Technologies*, vol. 100, pp. 68 – 87, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X19300282

[6] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "Depthcn: Vehicle detection using 3d-lidar and convnet," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–6.

[7] B. Coifman, M. Wu, K. Redmill, and D. A. Thornton, "Collecting ambient vehicle trajectories from an instrumented probe vehicle: High quality data for microscopic traffic flow studies," *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 254–271, 2016.

[8] P. Steinemann, J. Klappstein, J. Dickmann, H. . Wünsche, and F. v. Hundelshausen, "Determining the outline contour of vehicles in 3d-lidar-measurements," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011, pp. 479–484.

[9] H.-y. Wang and H.-C. Shih, "A robust vehicle model construction and identification system using local feature alignment," in *2013 IEEE International Symposium on Consumer Electronics (ISCE)*. IEEE, 2013, pp. 57–58.

[10] A. Azim and O. Aycard, "Detection, classification and tracking of moving objects in a 3d environment," in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 802–807.

[11] T. Miyasaka, Y. Ohama, and Y. Ninomiya, "Ego-motion estimation and moving object tracking using multi-layer lidar," in *2009 IEEE Intelligent Vehicles Symposium*, June 2009, pp. 151–156.

[12] H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, "Pedestrian recognition and tracking using 3d lidar for autonomous vehicle," *Robotics and Autonomous Systems*, vol. 88, pp. 71 – 78, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015302633

[13] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2d lidar," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1039–1063, 2015.

[14] P. Steinemann, J. Klappstein, J. Dickmann, F. von Hundelshausen, and H.-J. Wünsche, "Geometric-model-free tracking of extended targets using 3d lidar measurements," in *Laser Radar Technology and Applications XVII*, vol. 8379. International Society for Optics and Photonics, 2012, p. 83790C.

[15] J. Owens, A. Hunter, E. Fletcher *et al.*, "A fast model-free morphology-based object tracking algorithm," 2002.

[16] Z. Wang, X. Zhao, Z. Xu, X. Li, and X. Qu, "Modeling and field experiments on lane changing of an autonomous vehicle in mixed traffic," *Computer-aided Civil and Infrastructure Engineering*.

[17] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, 1998.

[18] Y. Malinovskiy, J. Zheng, and Y. Wang, "Model-free video detection and tracking of pedestrians and bicyclists," *Computer-Aided Civil and Infrastructure Engineering*, vol. 24, no. 3, pp. 157–168, 2009.

[19] V. LiDAR, *VLP-16 User's Manual and Programming Guide*, 2015.

[20] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3d laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 41–52, 2017.

[21] P. F. Lima, R. Oliveira, J. Mårtensson, and B. Wahlberg, "Minimizing long vehicles overhang exceeding the drivable surface via convex path optimization," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

Appendix A: Package Structure of Velodyne LiDAR Data

This document outlines package structure of Velodyne LiDAR and provides explanations on terminologies related to LiDAR data package. The Velodyne LiDAR data are stored in .pcap files where "pcap" stands for "packet capture". Hence, a .pcap file generally contains information captured from network traffic, which is commonly used in computer network administration. The Velodyne LiDAR, connected to a field computer, serves as a network device which communicates with the computer through ethernet. Therefore, the sampled data sends to the computer are captured and saved in .pcap data format.

Figure A.1 shows the raw data seen from WireShark, a network analysis tool. The data package are displayed in three sections. The top section shows all packages presented in this file with sequence ID and recorded time where total number of packages matches the frame length. IP addresses of both the LiDAR and the computer, communication protocol and package length are also reported, but they are not of our interests. The middle section displays detail information of individual packages, and the full raw data of a selected package, coded in hex number, are shown in the bottom section.

[19] provides package structure, as shown in A.2, needed to decode raw messages. The data package contains scan returns from each scan angle indexed with time. Each

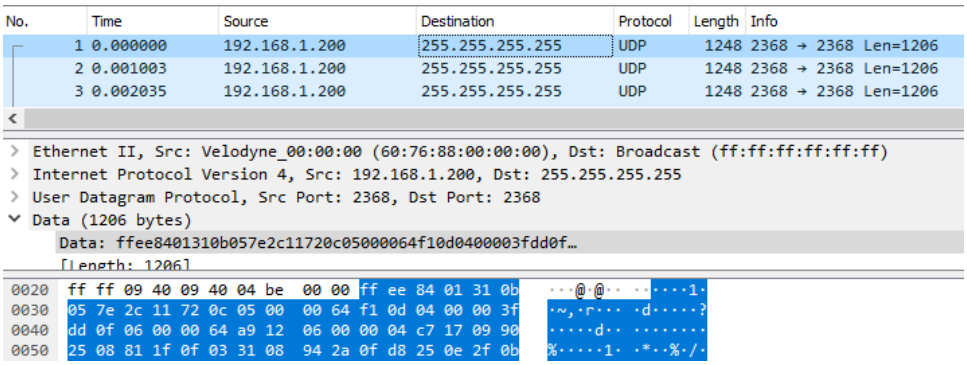

Figure A.1: Velodyne LiDAR data package as seen from network analysis tool

scan angle has a designated laser ID from 0 to 1, which matches scan angle -15° to 15° with 2° increment.

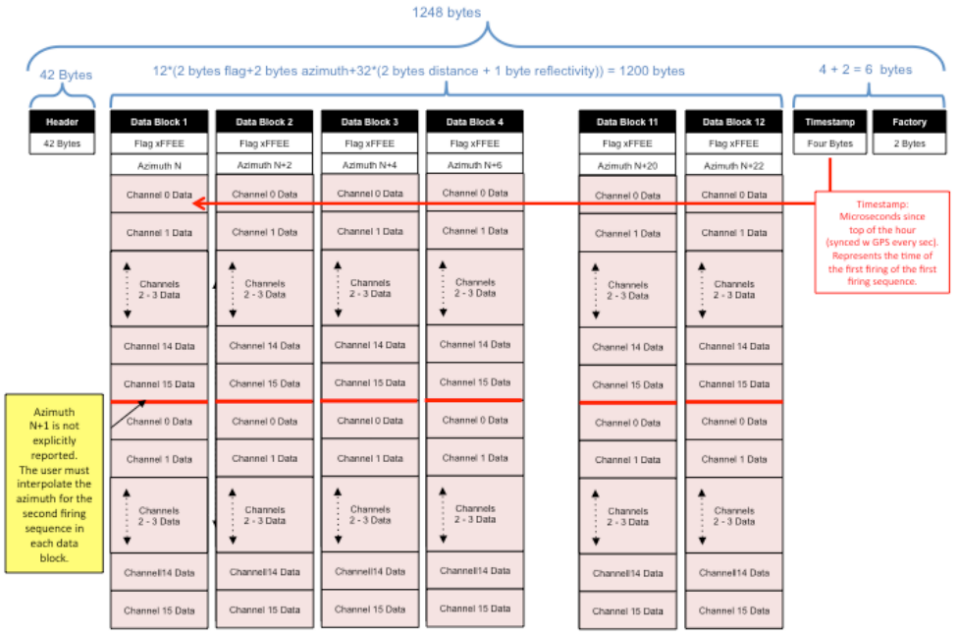

Figure A.2: Data package structure of Velodyne VLP16 LiDAR in single return mode [19]

Appendix B: Design of *LidarVisualizer* - a Point Cloud Visualization and Processing Tool

B.1 Introduction

In this report, we present the design of *LidarVisualizer* - a MATLAB-based Graphic User Interface application. Data visualization and pre/post-processing tools were designed, and the application is created on MATLAB App Designer. This design facilitates LiDAR point cloud visualization and manual processing where an application is not available currently to meet both needs. The application meets all objectives, following design process. The software is able to load, display and export a LiDAR image, and features manual segmentation and clustering operations. Although the toolkit is developed to assist the research in 3D LiDAR based vehicle tracking, it has flexibility to accommodate other processing and visualization needs for other LiDAR-based applications.

B.2 Design Specification

B.2.1 Preliminary Investigation

A visualization tool is needed to better understand the formulation of collected data. In our study, a VLP16 3D LiDAR, manufactured Velodyne, was used in our

study where road traffic data were collected. The collected data consists of multiple returns captured at different scan angles for a duration of 100 seconds. Currently, the only available tool to present the LiDAR is VeloView created jointly by Velodyne and ParaView. Despite the software is strong in displaying the data, it does not provide any interface to enable processing or connect to processing platform like MATLAB. For example, there is no way to visualize the processed data after the ground returns were labeled in a given frame. In this case, a new processing toolkit is necessary to support real-time visualization to facilitate result validation of LiDAR processing.

B.2.2 Parameter Gathering

Data structure of existing LiDAR data pack was studied to understand. The data is loaded into MATLAB by its built-in support to construct a data loader, which returns a point cloud object given a frame ID where the point cloud contain location information and sampled time stamp. Location information coded in the point cloud object follows technical specification of respected model as given by manufacturer[1]. In our case, the second dimension of returned location matrix is 16, which matches the 16 scan angle of VLP 16 LiDAR.

Besides the standalone data format standard by Velodyne, other industrial standards on LiDAR point cloud and 3D data format were studied. ASTM E57, regulated by ASTM Committee E57 on 3D Imaging Systems[2], is a standard committee regulates 3D data exchange format. The format consists a XML component which enabling potential interfacing with LiDAR data. Alternatively, storing a 3D point cloud can employ one of the Point Cloud Data (PCD) file format.

[2]VeloDyne LiDAR Manual: https://velodynelidar.com/downloads/

[2]ASTM E57: https://www.astm.org/COMMITTEE/E57.htm

Key features needed for general purpose data processing and visualization were collected. For visualization purpose, we uses orthographic views, given by ASME Y14.3-2003 standard[3], to display a 3D LiDAR image. We collected key features for point cloud processing based on applications in the field.

B.3 Evaluation of Design

B.3.1 Simulation and Prototyping

Figure B.1 shows the initial GUI layout of *LidarVisualizer* containing a control panel and a display panel. Test data were generated including randomly generated point cloud and were loaded to the application to test the basic display functionality. Simple geometry point clouds were generated in order to test geometric transformation feature of the application.

B.3.2 Testing and Demonstration

Sample LiDAR data was used to test the functionalities of the data. The initial implementation suggested additional features shall be added, including graphic input. Hence, the built-in display panel was removed, and all graphics are rendered on MATLAB figure. The *LidarVisualizer* was demonstrated in result validation stage where the LiDAR data containing ground, non-ground labels are color coded and properly displayed.

The application was presented in the group's weekly meeting and was distributed to an undergraduate researcher in our lab to perform manual data labeling. We gathered feedback from all researchers in our lab. The use of MATLAB App Designer

[3]ASME Y14.3-2003 standard: https://www.asme.org/codes-standards/find-codes-standards/y14-3-orthographic-pictorial-views

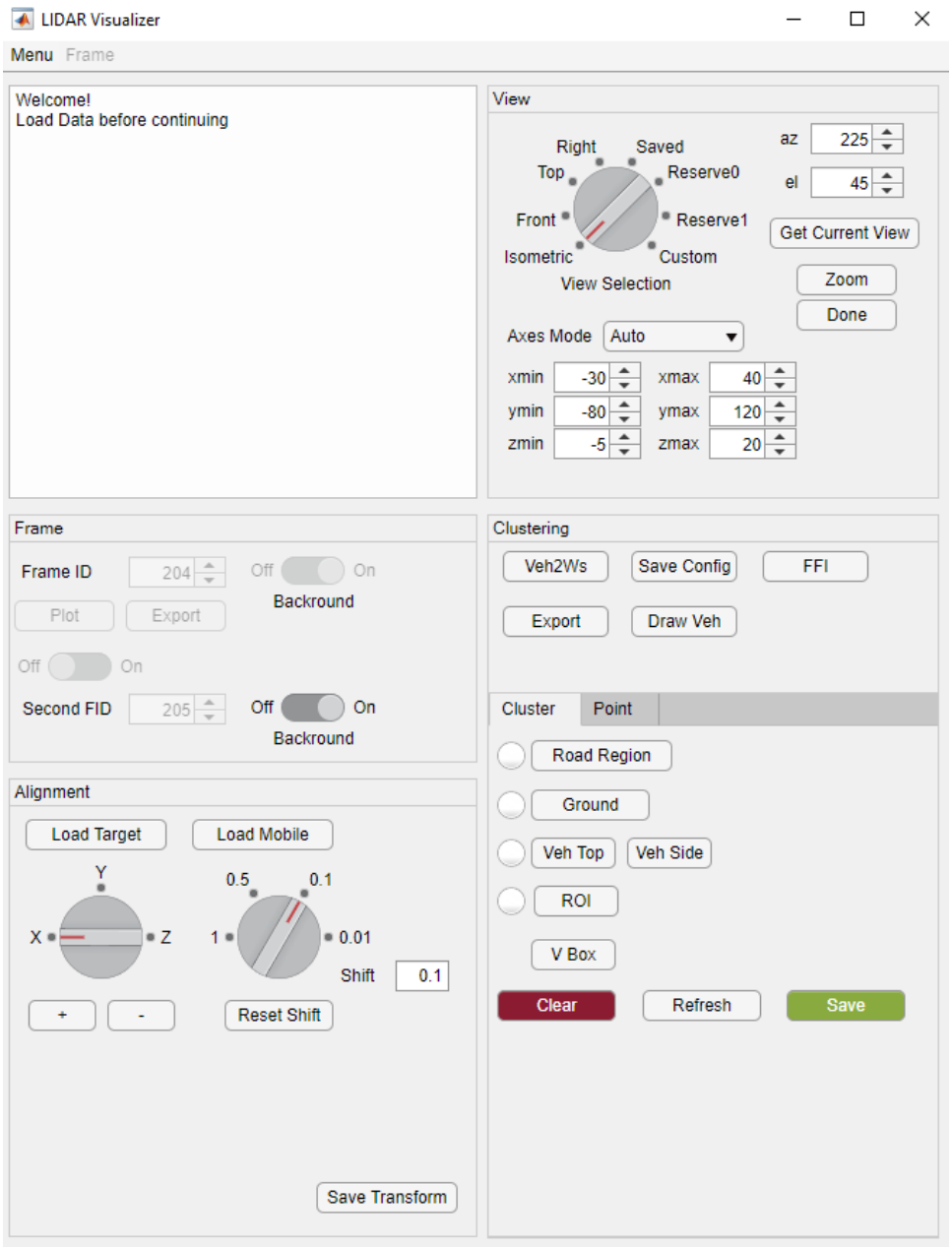

Figure B.1: LidarVisualizer application user interface

and data structure were discussed and shared with a graduate researcher designing a video data processing tool. The update was made based on the feedbacks where the size of exported data was optimized. The feedback from the undergraduate researcher

also suggests the software improves the proficiency of using the MATLAB since the application is open sourced.

B.4 Conclusion and Future Work

We reported design, implementation and testing of *LidarVisualizer*. The application features point cloud processing and visualization, providing seamless connection from raw data file to processing platform. The application will be integrated with additional features to facilitate data pre-processing, like initialize a rotation matrix and apply to the loaded data. Also, the software will enable Foreign Function Interface (FFI) to Python programming language, enabling calling existing algorithm that does not exist in MATLAB environment. Both improvements will further facilitate prototyping and enabling algo developing.